Progress

**Highlights**

1. Began working on methods section of ABM paper, almost finished with first draft
2. Proposed new distributed load-balancing strategy – distributed to Wenwu and Shaowen
3. Compared fractal surfaces with agent clustering (H=.3 and H=.7 - radius=512) **No difference in simulation time**
4. Proposed UML DB design for GISolve 2 job management – distributed to Yan and Anand
5. Started designing job management system

1. **Methods section**

A complete draft of the methods section of the ABM paper will be finished before the I leave for the thanksgiving break and will be distributed to Shaowen and Wenwu. I am almost finished with the first draft, hopefully I will finish one or two more revisions after the first draft.

2. **New load-balancing strategy**

I confirmed that Nichols 1994 strategy is breaking when scaling to a large number of processors. I tried a temporary fix, but it even broke under some conditions. I proposed an alternative distributed load-balancing strategy and sent the short writeup to Wenwu and Shaowen. Wenwu said he scanned over it and it looked good so far. I will begin implementation today or tomorrow.

3. **Fractal surface comparison**

As requested during the last ABM meeting I compared two sets of fractal surfaces (H=.3 and H=.7) with agent clustering (radius=512). The comparison yielded the same average iteration time (32.86 seconds) for 4096x4096 environment size and 16 million agents on 256 processors. I discussed the results with Wenwu and will be comparing the fractal surfaces with uniform agent distribution instead of clustered distribution.

4. **UML diagram for GISolve2 job management system**

A version of this was proposed to Yan and Anand previously (see: [https://www.cigi.uiuc.edu/doku.php/projects/gisolve2/jobman](https://www.cigi.uiuc.edu/doku.php/projects/gisolve2/jobman)). This version has only a minor modification that is noted in Graph 1. Notice that this diagram depends on DB tables that are outside the scope of job management. I am waiting on the schema from the other tables to finalize this diagram.

5. **Job management system**

I started work on designing the components for the GISolve2 job management system. I showed an initial component list to Anand and he suggested to start coding immediately. I started programming a few of the components and will continue development in the coming weeks. The proposed system will use independent components that will directly interact with the DB, such that each component could be replaced without affecting the other components. The components will be developed in perl, because of its DB and scripting capabilities. There are some unresolved issues in dealing with the diverse grid systems (i.e. Teragrid and OSG) that I am still working out. A preliminary component list is included below.
Results and discussion

UML diagram for GISolve2 job management system

Graph 1: This is the proposed UML diagram for the job submission system. The diagram above is a new version and has moved local-data outside of job management, because local data should be the responsibility of the core of GISolve2 since it will have to manage user permissions for the data and have access to it for export and visualization. Therefore I moved local-data to be outside the scope of job management.

Job submission system component list

A preliminary component list with brief description
1. Job submit
   submit a job – return grid id
2. Data copy
   copy data to/from remote site based on rdataid
3. Job status check
   given job id – return status of job
4. Data status check
   given rdata id – return status of data
5. Match maker
   given job id, return matched site id
6. Run maker
   given experiment, create all runs and jobs
7. Data associator
   given experiment, runs, and jobs
   associate data and make assoc-data entries
8. Tracker
   Track progress of runs (and jobs) using job/data status check
   if dependencies for next stage are met
   trigger next stage operation (job submit or data copy)
9. Site update
   Given site id, update status
   Status could be availability, approximate queue time, number of open job slots, etc
10. Data maker
    given experiment, run, and job information
    pass this information to a custom script that makes data such as input / parameter files
    return location of made files?
    Used for generating custom data for specific applications

This is a preliminary list. Any comments or criticisms are welcomed to help speed development.